02986.P045

UNITED STATES PATENT APPLICATION

FOR

CIRCUITS WITH MODULAR REDUNDANCY AND METHODS AND APPARATUSES FOR THEIR AUTOMATED SYNTHESIS

INVENTOR:

KENNETH S. MCELVAIN

PREPARED BY:

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN
12400 WILSHIRE BOULEVARD
SEVENTH FLOOR
LOS ANGELES, CA 90025-1026

(408) 720-8598

CIRCUITS WITH MODULAR REDUNDANCY AND METHODS AND APPARATUSES FOR THEIR AUTOMATED SYNTHESIS

[0001] This application is a Continuation-In-Part (CIP) of U.S. Patent Application Serial No. 10/407,678, entitled "Method and Apparatus for Automated Synthesis of Multi-channel Circuits", filed April 4, 2003 by Levent Oktem and Kenneth S. McElvain.

FIELD OF THE TECHNOLOGY

[0002] The field relates to digital circuits, and more particularly to circuits with modular redundancy and their automated design.

BACKGROUND

[0003] Redundant components have been used to improve the reliability of a system. For example, Triple Modular Redundancy (TMR) has been used to improve the reliability of a digital processing system, in which a voting circuit compares the redundant results of three redundant circuits that simultaneously perform the same computation to migrate Single Event Upsets (SEU). For example, two-out-of-three voting has been used to elect the final result and eliminate the effect of SEU.

[0004] TMR can be used at different levels (e.g., at a module/device level, or at a gate level). For example, an entire module or device may be replicated to generate redundant instances of the module or device; and, the outputs of the redundant instances are compared to each other to vote for the final result. To limit the error propagation, the voting may also be performed at a gate level (e.g., before or after

sequential elements) so that the error in one section of a circuit will not propagate into another section of the circuit. For example, **Figure 26** illustrates one TMR approach which triplicates a D-type flip-flop to create redundant instances (e.g., flip-flops 2011, 2013, and 2015) and applies a voter (e.g., voter 2002, which includes AND gates 2021, 2023 and 2025 and OR gate 2027) to process the simultaneous outputs of the redundant instances of the D-type flip-flop to generate a TMR protected output (e.g., 2003). By replacing each of the flip-flops in the design with the redundant instances and the voter (e.g., replacing each flip-flop with an instance of circuit 2000), the design is protected against SEUs occurred in the flip-flops.

[0005] For the design of digital circuits (e.g., on the scale of Very Large Scale Integration (VLSI) technology), designers often employ computer-aided techniques. Standard languages such as Hardware Description Languages (HDLs) have been developed to describe digital circuits to aid in the design and simulation of complex digital circuits. Several hardware description languages, such as VHDL and Verilog, have evolved as industry standards. VHDL and Verilog are general-purpose hardware description languages that allow definition of a hardware model at the gate level, the register transfer level (RTL) or the behavioral level using abstract data types. As device technology continues to advance, various product design tools have been developed to adapt HDLs for use with newer devices and design styles.

[0006] In designing an integrated circuit with an HDL code, the code is first written and then compiled by an HDL compiler. The HDL source code describes at some level the circuit elements, and the compiler produces an RTL netlist from this compilation. The RTL netlist is typically a technology independent netlist in that it

is independent of the technology/architecture of a specific vendor's integrated circuit, such as field programmable gate arrays (FPGA) or an application-specific integrated circuit (ASIC). The RTL netlist corresponds to a schematic representation of circuit elements (as opposed to a behavioral representation). A mapping operation is then performed to convert from the technology independent RTL netlist to a technology specific netlist, which can be used to create circuits in the vendor's technology/architecture. It is well known that FPGA vendors utilize different technology/architecture to implement logic circuits within their integrated circuits. Thus, the technology independent RTL netlist is mapped to create a netlist, which is specific to a particular vendor's technology/architecture.

SUMMARY OF THE DESCRIPTION

[0007] Digital circuits with time multiplexed redundancy and methods and apparatuses for their automated designs generated from single-channel circuit designs are described here. Some embodiments of the present invention are summarized in this section.

[0008] At least one embodiment of the present invention includes a digital circuit which detects or corrects transitory upsets through time-multiplexed resource sharing. In one embodiment of the present invention, time-multiplexed resource sharing is used to reduce the die area for implementing modular redundancy. One embodiment of the present invention automatically and efficiently synthesizes multi-channel hardware for time-multiplexed resource sharing by automatically generating a time-multiplexed design of multi-channel circuits from the design of a single-channel circuit, in which at least a portion of the channels are allocated for modular redundancy.

[0009] In one embodiment of the present invention, a digital circuit with redundancy protection, includes: a time multiplexer to assign a plurality of redundant data into a plurality of time slots; a time-multiplexed circuit, coupled to the time multiplexer, to process the plurality of redundant data in the plurality of time slots to generate a plurality of redundant results respectively; and a voting circuit, coupled to the time-multiplexed circuit, to process the plurality of redundant results to maintain data integrity. In one example, the voting circuit determines an output result according to a majority of the redundant results and identifies a faulty

one of the plurality of redundant results. In one example, the time-multiplexed circuit has a plurality of state memory elements to store a plurality of redundant states; and, a reload logic circuit, coupled to the voting circuit and the time-multiplexed circuit, copies data from a first one of the plurality of state memory elements which corresponds to the majority of the redundant results to a second one of the plurality of state memory elements which corresponds to the faulty one of the plurality of redundant results. In one example, the voting circuit determines, in the plurality of time slots, a plurality of voting results based on the plurality of redundant results. In one example, the time-multiplexed circuit includes a combinatorial logic circuit. In one example, the time-multiplexed circuit pipelines processing of the plurality of redundant data. In one example, the plurality of time slots correspond to a plurality of continuous clock cycles for the time-multiplexed circuit; the plurality of redundant data is for a first channel; the time multiplexer assigns a plurality of redundant data for a second channel into time slots after the plurality of time slots for the first channel; and, data for the first channel is independent from data for the second channel. Alternatively, the plurality of time slots correspond to a plurality of discontinuous clock cycles for the time-multiplexed circuit; and, the plurality of redundant data is for a first channel; the plurality of time slots are separated by a plurality of time slots for a plurality of redundant data for a second channel; and, data for the first channel is independent from data for the second channel. In one example, the time-multiplexed circuit includes a plurality of pipelined registers to pipeline an intermediate result for the plurality of time slots. In one example, the digital circuit is integrated on one chip.

[0010] In another embodiment of the present invention, a method to design a digital circuit with redundancy protection, includes: automatically generating a second design of a time multiplexed circuit from a first design of a single-channel circuit, where the time multiplexed circuit is configured to process a plurality of redundant data in a plurality of time slots to generate respectively a plurality of redundant results; and generating a voting circuit to process the plurality of redundant results to maintain data integrity. In one example, the voting circuit determines an output result according to a majority of the redundant results; and, the voting circuit identifies a faulty one of the plurality of redundant results. In one example, the time-multiplexed circuit has a plurality of state memory elements to store a plurality of redundant states; and a reload logic circuit is further generated to copy data from a first one of the plurality of state memory elements which corresponds to the majority of the redundant results to a second one of the plurality of state memory elements which corresponds to the faulty one of the plurality of redundant results. In one example, the voting circuit determines, in the plurality of time slots, a plurality of voting results based on the plurality of redundant results. In one example, the time-multiplexed circuit includes a combinatorial logic circuit. In one example, a time multiplexer is further generated to assign the plurality of redundant data into the plurality of time slots; and, the time-multiplexed circuit pipelines processing of the plurality of redundant data. In one example, the plurality of time slots correspond to a plurality of continuous clock cycles for the time-multiplexed circuit; the plurality of redundant data is for a first channel; the time multiplexer assigns a plurality of redundant data for a second channel into time slots

after the plurality of time slots for the first channel; and, data for the first channel is independent from data for the second channel. Alternatively, the plurality of time slots correspond to a plurality of discontinuous clock cycles for the time-multiplexed circuit; the plurality of redundant data is for a first channel; the plurality of time slots are separated by a plurality of time slots for a plurality of redundant data for a second channel; and, data for the first channel is independent from data for the second channel. In one example, the time-multiplexed circuit includes a plurality of pipelined registers to pipeline an intermediate result for the plurality of time slots. In one example, generating the second design includes: generating a multi-state Finite-State-Machine (FSM) to time multiplex access to logic elements of the first design by multiple channels according to time slots which include the plurality of time slots for the plurality of redundant data. In one example, generating the second design further includes: generating a multiplexing circuit to time multiplex multiple inputs for the multiple channels onto an input line of the first design, where the multiple inputs include the plurality of redundant data. In one example, generating the second design further includes: replacing a channel-specific element (e.g., a constant; a Random Access Memory (RAM) element; a Read Only Memory (ROM) element; a register; a flip-flop; and a negative latency register) in the first design with multiple corresponding elements, each of the multiple corresponding elements being accessed for one of the multiple channels according to a state of the FSM. In one example, the channel-specific element is a channel-specific sequential element. In one example, non-channel-specific sequential elements (e.g., a set of pipeline register) are identified. In one example, the channel-specific sequential element is replaced with a

cascade of multiple shifting sequential elements. In one example, the channel-specific sequential element is replaced with multiple memory elements addressed according to the state of the FSM. In one example, a number of feed-forward cutsets of sequential elements are determined as non-channel-specific sequential elements. In one example, generating the second design further includes: replacing a sequential element (e.g., a constant; a Random Access Memory (RAM) element; a Read Only Memory (ROM) element; a register; a flip-flop; and a negative latency register) in the first design with corresponding elements to generate the second design, where the corresponding elements are sequentially accessed in the second design according to timing for processing signals from multiple channels. In one example, the corresponding elements are addressed sequentially. In one example, generating the second design includes: generating a conglomerate of single-channel circuits of the first design; and applying a folding transformation to the conglomerate of single-channel circuits to generate the second design of the time multiplexed circuit. In one example, information indicating a parallelism (e.g., information of a folding set) in the conglomerate of single-channel circuits is further generated; and, the folding transformation uses the information indicating the parallelism to generate the second design.

[0011] The present invention includes methods and apparatuses which perform these methods, including data processing systems which perform these methods, and computer readable media which when executed on data processing systems cause the systems to perform these methods.

[0012] Other features of the present invention will be apparent from the

accompanying drawings and from the detailed description which follows.

BRIEF DESCRIPTION OF THE DRAWINGS

[0013] The present invention is illustrated by way of example and not limitation in the figures of the accompanying drawings in which like references indicate similar elements.

[0014] **Figure 1** shows a block diagram example of a data processing system which may be used with the present invention.

[0015] **Figure 2** shows an example of a single-channel three-tap Finite Impulse Response (FIR) filter from which a multi-channel filter can be automatically generated according to one embodiment of the present invention.

[0016] **Figure 3** shows signal waveforms at various locations in the example of **Figure 2**.

[0017] **Figure 4** shows an example of a two-channel filter corresponding to the conglomerate of single-channel filters of **Figure 2**.

[0018] **Figure 5** shows an example of a two-channel filter automatically generated from the single-channel filter of **Figure 2** according to one embodiment of the present invention.

[0019] **Figure 6** shows signal waveforms at various locations in the example of **Figure 5**.

[0020] **Figure 7** shows an example of a multi-channel filter automatically generated from the single-channel filter of **Figure 2** according to one embodiment of the present invention.

[0021] **Figure 8** shows an example of an output decoder and latch circuit for de-

multiplexing outputs from a multi-channel filter according to one embodiment of the present invention.

[0022] **Figure 9** shows another example of a multi-channel filter automatically generated from the single-channel filter of **Figure 2** according to an alternative embodiment of the present invention.

[0023] **Figure 10** shows an example of a single-channel three-tap Finite Impulse Response (FIR) filter with pipeline registers from which a multi-channel filter can be automatically generated according to one embodiment of the present invention.

[0024] **Figure 11** shows an example of a multi-channel filter automatically generated from the single-channel filter of **Figure 10** according to one embodiment of the present invention.

[0025] **Figure 12** shows another example of a single-channel circuit with pipeline registers from which a multi-channel filter can be automatically generated according to one embodiment of the present invention.

[0026] **Figure 13** shows an example of a multi-channel filter automatically generated from the single-channel filter of **Figure 12** according to one embodiment of the present invention.

[0027] **Figure 14** shows a flow chart of a method to generate a multi-channel circuit from a single-channel circuit according to one embodiment of the present invention.

[0028] **Figure 15** shows a detailed flow chart of a method to generate a multi-channel circuit from a single-channel circuit according to one embodiment of the present invention.

[0029] **Figure 16** shows an example method to generate a multi-channel circuit from a single-channel circuit according to one embodiment of the present invention.

[0030] **Figures 17 – 21** illustrate examples of generating a multi-channel circuit from a single-channel circuit with negative latency registers according to one embodiment of the present invention.

[0031] **Figures 22 – 25** illustrate another example of generating a multi-channel circuit from a single-channel circuit using negative latency registers according to one embodiment of the present invention.

[0032] **Figure 26** illustrates a prior art circuit of Triple Modular Redundancy (TMR) for protecting data in a flip-flop against Single Event Upsets (SEU).

[0033] **Figures 27 – 28** illustrate circuits with time-multiplexed redundancy for protection against transitory upsets according to embodiments of the present invention.

[0034] **Figure 29** illustrates a circuit with time-multiplexed redundancy for protection against transitory upsets in both computation logic and voting logic according to one embodiment of the present invention.

[0035] **Figure 30** illustrates a voting logic circuit with time-multiplexed redundancy for protection against transitory upsets according to one embodiment of the present invention.

[0036] **Figure 31** shows signal waveforms at various locations in the example of **Figure 30**.

[0037] **Figures 32 – 33** illustrate circuits with time-multiplexed redundancy for processing multiple independent channels with protection against transitory upsets

according to embodiments of the present invention.

[0038] **Figure 34** illustrates state memory elements for a time-multiplexed circuit which is capable of being reloaded to correct a faulty thread according to one embodiment of the present invention.

[0039] **Figures 35 – 36** illustrate circuits which are capable of reloading state memory elements of a faulty thread according to embodiments of the present invention.

[0040] **Figure 37** shows a method to design a circuit with time-multiplexed modular redundancy protection against transitory upsets according to one embodiment of the present invention.

[0041] **Figure 38** shows a detailed method to design a circuit with time-multiplexed modular redundancy protection against transitory upsets according to one embodiment of the present invention.

[0042] **Figure 39** shows a detailed method to design a time-multiplexed circuit for multiple independent channels with modular redundancy protection against transitory upsets according to one embodiment of the present invention.

DETAILED DESCRIPTION

[0043] The following description and drawings are illustrative of the invention and are not to be construed as limiting the invention. Numerous specific details are described to provide a thorough understanding of the present invention. However, in certain instances, well known or conventional details are not described in order to avoid obscuring the description of the present invention. References to one or an embodiment in the present disclosure are not necessary to the same embodiment; and, such references mean at least one.

[0044] At least one embodiment of the present invention seeks to detect or correct transitory upsets through time-multiplexed resource sharing. In one embodiment of the present invention, time-multiplexed resource sharing is used to reduce the die area for implementing modular redundancy. One embodiment of the present invention automatically and efficiently synthesizes multi-channel hardware for time-multiplexed resource sharing by automatically generating a time-multiplexed design of multi-channel circuits from the design of a single-channel circuit, in which at least a portion of the channels are allocated for modular redundancy. A subset of the channels that corresponds to the time slots assigned to perform redundant computation for a set of redundant data respectively is referred to as redundant threads in the present application.

[0045] Traditionally, Triple Modular Redundancy (TMR) has been used for protecting digital logic from Single Event Upsets (SEU) in space born application. For example, radiation may cause glitches in the digital circuit; and, redundant

modules can be used to improve the reliability of the system. As the industry advances and moves towards reduction in circuit device size (e.g., reducing the transistor size and the width of wires for an integrated circuit), the reliability of the circuit component in an operating environment may be decreased such that transitory upsets may become noticeable without redundancy protection. Radiation, voltage fluctuation, thermal noise and other environment conditions can cause transitory upsets, resulting erroneous computation results. Further, transitory upsets may occur not only in memory elements that store the data but also in combinatorial logic circuitry that performs the computation.

[0046] At least one embodiment of the present invention uses time-multiplexed resource sharing to reduce the die area in implementing TMR. A traditional Triple Modular Redundancy (TMR) method causes a significant increase in the die area in implementing the redundant instances. Driving the redundant instances simultaneously also increases the power consumption of the system. At least one embodiment of the present invention pipelines a number of redundant threads of computation using a time-multiplexed channel. Since a transitory upset affects only one of the redundant threads of computation, the same hardware can be used for implementing modular redundancy to protect against transitory upsets through time multiplexed resource sharing. Further, in one embodiment, the redundant threads are scheduled in continuous time slots so that there is no signal switching between redundant threads, if no upset appears; thus, the power consumption is significantly reduced, when such an embodiment is compared to the traditional TMR.

[0047] **Figure 27** illustrates a circuit with time-multiplexed redundancy for

protection against transitory upsets according to one embodiment of the present invention. Time multiplexer 2101 places redundant data from I^1 (2111), I^2 (2113) and I^3 (2115) onto input line 2127 at their respective time slots for time multiplexed multi-channel 2105. Time multiplexed multi-channel 2105 performs the redundant threads of computation using the redundant data to generate redundant results on line 2125 in different time slots. Demultiplexer 2103 decodes the time multiplexed redundant results on line 2125 according to their respective time slots into simultaneous inputs on lines 2112, 2114 and 2116 for voter 2107, which determines a majority of the inputs for voter 2107 as the output 2123. In **Figure 27**, voter 2107 is a traditional voter (e.g., voter 2002 illustrated in **Figure 26**), which takes three simultaneous redundant inputs to generate an output (e.g., using a two-out-of-three voting logic to remove a single upset). In one embodiment, time multiplexed multi-channel 2105 pipelines the processing of inputs in different time slots to have a fast processing speed; alternatively, the time multiplexed redundant inputs may be processed in a round-robin fashion. In one embodiment of the present invention, time multiplexed multi-channel 2105 starts to process each input in each clock cycle (e.g., according to clock signal C_3, which is three times faster than the clock signal for voter 2107); and, data for each thread of computation takes one or more clock cycles to propagate from input 2127 to output 2125. More details about embodiments of time multiplexed multi-channel 2105 and its automated design are described further below.

[0048] In **Figure 27**, a single circuit is used to generate redundant results in different time slots. Since a single transitory upset in the time multiplexed multi-

channel 2105 affects the result of only one of the redundant threads, the voting circuit can reliably detect and correct the upset. Since the redundant results are produced by processing the redundant data on the same hardware at different time slots, instead of redundant hardware, the die area required for the circuit is reduced (minimized). Further, in one embodiment, the redundant threads are pipelined in continuous clock cycles; thus, when there is no upset, there is no signal change between clock cycles for the redundant threads; and, the power consumption is significantly reduced.

[0049] In **Figure 27**, a single voter is used to generate the output result. Since the voter may also be subjected to transitory upsets, it may be also desirable to protect the voter against transitory upsets (since the output of the voter is the input of the next stage processing circuit). A traditional TMR approach can be used with the time-multiplexed channel to protect the voter. **Figure 28** illustrates such an embodiment. In **Figure 28**, redundant voters 2131, 2133 and 2135 process the outputs of demultiplexer 2103 to generate redundant results 2141, 2143 and 2145, which are redundant inputs for the next stage processing. Thus, a single upset in the voting circuitry (e.g., voters 2131, 2133 and 2135) can also be detected and corrected in the next stage processing.

[0050] **Figure 29** illustrates a circuit with time-multiplexed redundancy for protection against transitory upsets in both computation logic and voting logic according to one embodiment of the present invention. Instead of using three redundant voters, the example of **Figure 29** uses time-multiplexed voter 2155 for the generation of redundant voting results. One embodiment of the time-multiplexed

voter 2155 is illustrated in **Figure 30**. In **Figure 30**, registers R_1 (2201), R_2 (2203) and R_3 (2205) decode from V_i 2233 (through shifting) the time multiplexed redundant inputs in different time slots into the corresponding registers. When the redundant inputs are in registers R_1 (2201), R_2 (2203) and R_3 (2205) respectively, control signal C_L for multiplexers 2221, 2223 and 2225 becomes 1 to load the redundant inputs from registers R_1 (2201), R_2 (2203) and R_3 (2205) into registers R_4 (2211), R_5 (2213) and R_6 (2215) respectively. While registers R_1 (2201), R_2 (2203) and R_3 (2205) demultiplexing the next set of redundant inputs, R_4 (2211), R_5 (2213) and R_6 (2215) rotate the redundant inputs in these registers to provide inputs for voter 2231, which may be a traditional voter without sequential elements (e.g., voter 2002 shown in **Figure 26**). When a single voter design with one or more sequential elements is used, the methods of generating time-multiplexed multi-channel circuit from a single channel design, which are described further below, can be used to generate voter 2231 from the single voter design.

[0051] **Figure 31** shows signal waveforms at various locations in the example of **Figure 30**. At time t_0 (2241), register R_1 (2201) receives the first one of a set of redundant inputs, A_1^1. At time t_1 (2242), register R_1 (2201) receives the second one of the set of redundant inputs, A_1^2, and shifts A_1^1 into R_2 (2203). At time t_2 (2243), register R_1 (2201) receives the third one of the set of redundant inputs, A_1^3, and shifts A_1^2 into R_2 (2203), which shifts A_1^1 into R_3 (2205). Thus, just after time t_2 (2243), the set of redundant inputs A_1^1, A_1^2 and A_1^3 are demultiplexed into registers R_3, R_2 and R_1 respectively, as indicated by signals 2252, 2253 and 2254. At time t_3 (2244), control signal C_L is 1 (signal 2255); thus, the set of redundant inputs A_1^1, A_1^2

and A_1^3 are loaded into registers R_6, R_5 and R_4 respectively, as indicated by signals 2256, 2257 and 2258; and, voter 2231 generates result $V(A_1^1, A_1^2, A_1^3)$. At t_4 (2245) and t_5 (2246), control signal C_L is 0 (signal 2261 and 2263); and, the set of redundant inputs A_1^1, A_1^2 and A_1^3 are rotated within registers R_6, R_5 and R_4 to cause voter 2231 to generates results $V(A_1^2, A_1^3, A_1^1)$ and $V(A_1^3, A_1^1, A_1^2)$ respectively.

[0052] **Figure 32** illustrates a circuit with time-multiplexed redundancy for processing multiple independent channels with protection against transitory upsets according to one embodiment of the present invention. In **Figure 32**, time-multiplexed hardware 2301 and 2303 are not only used for redundant threads but also for independent channels. N independent channels use 3N time slots. The data for one of the independent channels is independent from the data for another one of the independent channels. Each of the independent channels has three redundant threads using three of the 3N time slots. In **Figure 32**, thread i (i=1,2,3) of channel j (j=1,2, ..., N) uses time slot $3(j-1)+i-1$; thus, the redundant threads for a channel use the continuous time slots such that, during the normal execution where there are no upsets, the signals remain the same across the clock cycles for the different redundant threads for the channel. Since the power consumption for a digital circuit is proportional to the amount of signal switching activities, the power consumption is reduced when the redundant threads use the adjacent time slots.

[0053] However, in some cases, a single transitory upset may last longer than one time slot (e.g., a single clock cycle of a time-multiplexed multi-channel). To improve the immunity of the circuit against such long transitory upsets, one may assign the adjacent time slots to different channels such that the redundant threads

are separated apart by a number of time slots. **Figure 33** illustrates such an example, in which thread i (i=1,2,3) of channel j (j=1,2, …, N) uses time slot $N(i-1)+j-1$. The redundant threads for a channel are separated by N−1 threads of other independent channels such that the circuit is immune to upsets that last up to N time slots (e.g., N clock cycles of the time multiplexed multi-channel), although the power consumption for such a digital circuit is typically much higher than the one illustrated in **Figure 32** due to the increased signal switching activities.

[0054] In one embodiment of the present invention, a voting circuit further identifies the faulty thread; and, a reload logic circuit is used to correct the data in the state memory elements for a faulty thread. **Figure 34** illustrates a set of state memory elements for a time-multiplexed circuit which is capable of being reloaded to correct a faulty thread according to one embodiment of the present invention. In a time multiplexed channel, a state is stored into a number of redundant state elements (e.g., 2411, 2413 and 2415); and, the redundant state elements are accessed according to the time slots assigned for the corresponding threads. For example, state elements 2411, 2413 and 2415 sample the input (2401) at the corresponding timing slots according to write control signals T_1 (2441), T_2 (2443) and T_3 (2445) during a write cycle; and, multiplexer 2403 outputs the redundant states stored in the state elements 2411, 2413 and 2415 onto line 2405 in the corresponding timing slots. Multiplexers 2421, 2423 and 2425 are used for the reload of a possible faulty state element from a normal state element.

[0055] In one embodiment, the reloading of the state elements are delayed until a cycle where the state is not being written into the state elements, in which cycle the

control signals D_1 (2431), D_2 (2433) and D_3 (2445) and T_1 (2441), T_2 (2443) and T_3 (2445) are selectively controlled to reset the faulty element to a normal one. For example, when element 2413 is faulty, signal D_2 is switched to 0 to load data from element 2415 during a non-write cycle.

[0056] In another embodiment, the reloading can also be performed during a write cycle for the state. For example, during a write cycle, the timing of the write control signal (e.g., T_1 (2441), T_2 (2443) and T_3 (2445)) for a faulty thread is temporality synchronized with the timing of a normal thread to obtain the input from the normal thread. For example, if the thread for element 2411 is faulty, signal T_1 (2441) is temporarily replaced with signal T_2 (2443) to sample the input from line 2401 during the write cycle. Thus, elements 2411 and 2413 will have the same value after the write cycle.

[0057] Further, in one embodiment, during the reload cycle, multiplexer 2403 may be controlled to block the data from the faulty thread. For example, the control signal for multiplexer 2403 may be modified, in response to the identification of a faulty thread, to replace the data from the element corresponding to a faulty thread with the data from the element of a normal thread. For example, when the thread corresponding to element 2411 is faulty, the multiplexer is controlled to select the data from element 2413 during the time slot 2, instead of from element 2411.

[0058] **Figure 35** illustrates a circuit which is capable of reloading state memory elements of a faulty thread according to embodiments of the present invention. In **Figure 35**, voter 2503 identifies the faulty thread when a single upset occurs. Since the faulty result may be due to an upset in the memory blocks 2507 of time

multiplexed multi-channel 2501, the memory block of the faulty thread is reloaded from the memory block of a normal thread in multiple clock cycles (according the size of the memory block) in one embodiment. Typically, a number of cycles passes after the reading of the memory block (e.g., 2507) and before the voter 2503 detects an error, the address that was used to access the memory block for producing the faulty result may be lost. Thus, in one embodiment, the entire memory block for a faulty thread is reloaded. Alternatively, an additional circuitry may be added to store the addresses for the number of cycles so that, when a faulty data is propagated from the memory block to the voter, the address of the possible faulty data can be identified for reload.

[0059] To simplify the reload logic for memory blocks, the memory blocks are locally protected against transitory upsets in one embodiment of the present invention, as illustrated in **Figure 36**. In **Figure 36**, redundant memory blocks 2525 are locally protected with a voter (e.g., 2521) so that a simple reload logic circuit (e.g., 2523) can be used to reload the faulty memory element. Since the error due to the upset is identified immediately after the reading, reloading logic can be greatly simplified. Further, in some application, the data in the memory blocks 2525 may persist for a long period of time without being used. To protect the content in such memory blocks against the accumulation of upsets over a long period of time, additional circuitry (not shown in **Figure 36**) is added to periodically examine the content of the memory block to ensure data integrity.

[0060] From this description, a person skilled in the art can envision many different variations of reloading schemes and circuits for the redundant state

elements that are accessed according to time slots. Although the above example is illustrated with a flip-flop type of register, it will be understood that these methods can also be used for other types of sequential elements (e.g., RAM memory) with or without various modifications. For example, redundant states may be stored in three RAM memory units; and, the entire RAM unit of a faulty thread may be reloaded in a number of clock cycles, when a faulty thread is identified. Further, redundant states may be stored in one dual-port RAM unit, where during a write cycle one write-port is used to write the data for the normal thread and the other for reloading the faulty thread. Further, other reload schemes may also be used, such as pausing the computation to reload (although it may cause loss of time and data in some applications).

[0061] Although the above examples are illustrated with one-input-one-output channels, from this description, it will be apparent to one skilled in the art that these embodiments of the present invention can also be applied to computational circuitry with multiple inputs and multiple outputs. Some examples are illustrated with one stage voting (e.g., **Figures 29** and **32**), the multiple time-multiplexed voting can also be used for multiple stages of computation (e.g., using a chain or a network of time-multiplexed computing circuits and voting circuits). Time multiplexed redundancy can be used at various levels, such as at a module or device level, or at a gate level, or in combinations.

[0062] **Figure 37** shows a method to design a circuit with time-multiplexed modular redundancy protection against transitory upsets according to one embodiment of the present invention. After operation 2601 receives a design of a

portion of a circuit (e.g., a single channel, a filter, a computation logic, a module, or a device), operation 2603 automatically transforms the portion of the circuit into a time multiplexed multi-channel design with time multiplexed redundant channels for upset detection and/or migration. In one embodiment, a time multiplexed multi-channel is automatically generated using one of the methods as illustrated in **Figures 2 – 25**; and, a voting circuit is automatically added. In one embodiment, redundant instances are first generated (e.g., with information about the parallelism to guide the a folding transformation); and, a folding transformation is then applied to generate the time-multiplexed multi-channel. In one embodiment of the present invention, the portion of the circuit is automatically partitioned into stages, where the time multiplexed redundant results of each of the stages are voted to limit the propagation of the effect of an upset.

[0063] **Figure 38** shows a detailed method to design a circuit with time-multiplexed modular redundancy protection against transitory upsets according to one embodiment of the present invention. Operation 2621 receives a design of a single-channel circuit. Operation 2623 generates a three-channel circuit that time multiplexes access to the logic elements of the single-channel circuit for redundancy processing. In one embodiment, the three channels of the circuit share a portion of the computation logic through time multiplexing; and, the computations of the three channels are pipelined in the circuit according to time slots assigned to the channels. Operation 2625 generates a voting circuit for detecting and removing a faulty one of the outputs of the three-channel circuit. In one embodiment, the voting circuit is also time multiplexed to generate redundant results according to the time slots for

the redundant channels. If operation 2627 determines that the single-channel circuit has a memory element, operation 2629 generates reload logic for reloading memory elements of the three-channel circuit when an upset is detected. In one embodiment, the inputs to the three-channel circuit are from upstream voters such that the reload logic is generated only for the reloading of the three-channel circuit. Registers for storing intermediate results are typically naturally reloaded from their upstream computation logic. Thus, it is not necessary to reload these registers. However, in some application, it is desirable to reload a faulty thread within a given number of cycles; thus, some of these registers can be selectively reloaded to reduce the number of cycles to reload a thread. In one embodiment of the present invention, the inputs to the three-channel circuit are from upstream sequential elements without intervening voters, in which case a reload logic circuit is also generated to reload these upstream sequential elements when a faulty thread is detected.

[0064] **Figure 39** shows a detailed method to design a time-multiplexed circuit for multiple independent channels with time-multiplexed modular redundancy protection against transitory upsets according to one embodiment of the present invention. Operation 2641 receives a design of a single-channel circuit. Operation 2643 generates a 3N-channel circuit that time multiplexes access to the logic elements of the single-channel circuit for redundancy processing of N independent channels. Operation 2645 distributes redundant channels in the time multiplexed 3N channels (e.g., assign redundant channels to continuous time slots for reduced power consumption or assign independent channels in continuous time slots to separate redundant channels for better immunity to upsets). The data for the independent

channels are normally different for different independent channels; the data in the redundant channels for one independent channel are normally the same unless there is an upset; and, the redundant channels for the independent channel perform redundant threads of computation for the independent channel. Operation 2647 generates a voting circuit for detecting and migrating a faulty result in one of the redundant channels. In one embodiment, the voting circuit is also time multiplexed to generate redundant voting results according to the time slots for the redundant channels. Operation 2649 generates reload logic for reloading state memory elements of a faulty channel from the corresponding memory elements of a redundant channel when the voting circuit identifies the faulty channel. In one embodiment, the reload logic further reloads the upstream memory elements of the 3N channel circuit when the voter detects a faulty thread.

[0065] Although the above examples are illustrated with a circuit of three redundant time multiplexed threads for an independent channel, a different number of redundant threads can also be used. For example, when two redundant threads are used, a voter can detect the error when a single upset occurs without identifying a correct result. In some applications which use block oriented computation, the block may simply be reprocessed on detection of an error. In this case identifying a correct result is not necessary and comparing two threads for equality is sufficient. Once an error is detected, a restart logic circuit can be used to restart the computation and generate a correct result. In one embodiment, a restart logic circuit includes memory to store states for restart the computation and computation logic to control the restarting of the computation. In another embodiment, the system in

which the circuit embedded has the capability of restarting the computation (e.g., when the system includes a controlling processor). Thus, a restart logic circuit can be simply a signal line, carrying an error signal generated by the voter when the results of different threads do not match, to activate the restart of the computation (e.g., through providing a restart signal to the controlling processor). Alternatively, more than three threads can be used to improve detection and correction probabilities.

[0066] Detailed methods for the automatic generation of time multiplexed multi-channel circuits are described below.

[0067] In designing a circuit, transformations are frequently performed to optimize certain design goals. For example, transformations may be performed to reduce the area used by a circuit. The folding transformation is one of the systematical approaches to reduce the silicon area used by an integrated circuit. By executing multiple algorithm operations on a single function unit, the number of functional units in the implementation can be reduced. More details about folding transformations can be found in "VLSI digital signal processing systems: design and implementation", by Keshab K. Parhi, Wiley-Interscience, 1999.

[0068] Time multiplexed resource sharing has been used in the digital circuitry. For example, Peripheral and Control Processors (PACPs) of the CDC 6600 computer, described by J. E. Thornton in "Parallel Operations in the Control Data 6600", AFIPS Proceedings FJCC, Part 2, Vol. 26, 1964, pp. 33-40, share execution hardware by gaining access to common resources in a round-robin fashion. Another example about resource sharing for multi-channel filters can be found in: Jhon J.

Leon Franco, Miguel A. Melgarejo, "FPGA Implementation of a Serial Organized DA Multichannel FIR Filter", Tenth ACM International Symposium on Field Programmable Gate Arrays, Monterey, California, February 24-26, 2002.

[0069] A conventional folding algorithm can be used to automatically generate a design with time-multiplexed resource sharing from a given design. A conventional folding algorithm identifies the multiple algorithm operations that can be time multiplexed to a single functional unit to reduce the number of functional units (e.g., adders, multipliers). However, given a Digital Signal Processing (DSP) design, a conventional folding algorithm spends a significant amount of time in extracting parallelism and dependencies and in optimizing computation schedules. The complexity of hardware synthesis grows super-linearly with the number of logic units involved in the computation. Thus, the larger the designs, the harder it is to optimize and transform the circuitry.

[0070] The complexity of hardware synthesis grows super-linearly with the number of logic units involved in the design. A multi-channel circuit typically has independent and identical single channels. An N-channel circuit typically requires N times more logical (computational) units than its single-channel counterpart in optimizing the circuit during hardware synthesis. When a conventional folding transformation is applied on the multi-channel circuit to generate a time multiplexed circuit in order to reduce the function units used, the conventional folding algorithm spends a significant amount of time in extracting parallelism and dependencies and in optimizing computation schedules. Thus, hardware synthesis optimization for time-multiplexed resource sharing is typically computationally intensive,

particularly for large designs.

[0071] According to one embodiment of the present invention, the synthesis optimization for time-multiplexed resource sharing for multiple channels is based on the design of a single channel. The computation for extracting parallelism and dependencies is avoided; and, a design with time-multiplexed resource sharing can be generated from the design of a single channel without the extensive computation of the conventional folding transformation. In one example, a single-channel design is transformed into a time-multiplexed N-channel design by replacing channel specific elements of the single-channel design (e.g., registers and memories) with their counterparts having N-times more capacity to pipeline the signal processing for multiple channels. More examples and details are described below.

[0072] **Figure 2** shows an example of a single-channel three-tap Finite Impulse Response (FIR) filter from which a multi-channel filter can be automatically generated according to one embodiment of the present invention. Single-channel circuit 200 receives input on line 201 to generate output on line 203. The single-channel design includes constants 211, 213 and 215, multipliers 221, 223 and 225, adders 235 and 237, and registers (flip-flops) 231 and 233. Input on line 201 is multiplied by multipliers 221, 223 and 225 with constants 211, 213 and 215; and, adders 235 and 237 sum the results to generate output on line 203. Clock signal C_S on line 205 controls the pipelining and the synchronization of the processing in the single-channel circuit.

[0073] **Figure 3** shows signal waveforms at various locations in the example of **Figure 2**. At time t_0 (301), input signal I_1 arrives on line 201. At time t_1 (303),

register 231 outputs the result of multiplier 221 to generate signal S_A 313 on line A 207. At time t_2 (305), register 233 outputs the result of adder 235 to generate signal S_B 315 on line B 209, which is the sum of signal S_A 313 and the output of multiplier 223. At time t_3, adder 237 sums signal S_B 315 and the output of multiplier 225 to generate signal O_1 317 on output line 203.

[0074] **Figure 4** shows an example of a two-channel filter corresponding to the conglomerate of single-channel filters of **Figure 2**. For example, circuits 401 and 403 have the same elements and structure as circuit 200 in **Figure 2**. The conventional solutions for the automatic synthesis of hardware systems do not handle multi-channel systems specially. The conventional methods treat a multi-channel design as a large conglomerate of single-channel systems. A conventional method typically presents the two-channel filter as shown in **Figure 4** without any indication of the inherent high level parallelism in the design. Thus, a hardware synthesis software program performs the computational intensive task of extracting parallelism and dependencies. To generate a design with time-multiplexed resource sharing, a folding transformation may be applied. Thus, the conventional method is computational intensive. Further, when a multi-channel system is interpreted as a conglomerate of single-channel systems, the inherent high level parallelism can be missed, resulting in spending much more computational resources in applying folding transformations and/or failure to sufficiently optimize the given design.

[0075] In one embodiment of the present invention, information related to the parallelism in a conglomerate of single-channel systems, which is automatically generated from the single channel design, is also generated to guide a general

folding algorithm to apply folding transformation in automatically transforming the a conglomerate of single-channel systems into a time multiplexed circuit. For example, in a typical folding algorithm (e.g., those described in "VLSI digital signal processing systems: design and implementation", by Keshab K. Parhi, Wiley-Interscience, 1999, pp. 149-187, and others), it is a time consuming operation to identify folding sets. A folding set is a set of operations that is to be executed by a same functional unit through time multiplexing. When the conglomerate of single-channel systems is automatically generated from the single channel design, the folding sets can be generate without extensive computation. Such identified folding sets can be used to guide a typical folding algorithm in transforming the conglomerate of single-channel systems (e.g., traditional folding algorithms can be modified to use such information for efficient operations).

[0076] Instead of applying folding transformations to a large conglomerate of single-channel systems, at least one embodiment of the present invention explores fully the parallelism in the multi-channel circuits by applying automated transformations on the single-channel design to generate a design that enables time multiplexed resource sharing.

[0077] One embodiment of the present invention presumes multi-channel behavior from the beginning to avoid spending time in extracting parallelism and dependencies and in optimizing computation schedules, etc. Taking the single-channel version of the system as the input, it applies a simple mapping to make very effective use of the high level parallelism inherent in the multi-channel system. Since the data flows within the separate threads of the resulting multi-channel design

are identical, the resulting control circuitry is minimal.

[0078] **Figure 5** shows an example of a two-channel filter automatically generated from the single-channel filter of **Figure 2** according to one embodiment of the present invention. Functional units (e.g., multipliers and adders) are not channel specific. Since constants C1, C2, and C3 (211, 213 and 215) are not channel specific, they can also be shared without modification. Registers are in this case channel specific. Thus, registers 231 and 233 of single-channel design 200 are replaced with cascades of 2-shift registers 531, 533 and 535, 537. Inputs 501 and 503 (e.g., corresponding to inputs 411 and 413 of channel 401 and 403 in **Figure 4**) are time multiplexed by multiplexer 541 onto input line 505 according to signal S_M (507); and, output 509 is de-multiplexed by de-multiplexer 543 onto lines 513 and 515 (e.g., corresponding to outputs 421 and 423 of channel 401 and 403 in **Figure 4**). It is understood that the generation of multiplexer 541 (or de-multiplexer 543) is optional, depending whether or not the multiple-channel receives (or generates) signals on multiple ports.

[0079] **Figure 6** shows signal waveforms at various locations in the example of **Figure 5**. Input signals I_1 (615) and I_2 (617) are time multiplexed as signal S_I according to the state of control signal S_M (613). In a typical clock cycle of the single-channel circuit (e.g., from time t_0 301 to t_1 303 in **Figure 3**, which correspond to time t_0 601 to t_2 603 in **Figure 6**), each input signal is assigned to one time slot (e.g., slot 621 for signal I_1 615 and slot 622 for signal I_2 617). The input signal assigned for a given slot is processed by the logic units and pipelined by the cascades of registers for further processing. For example, signal I_1 615 assigned to

slot 621 in S_I is multiplied by constant 211 to generate intermediate result 631 in S_{A1} at time t_1 on line A1 (521) from register 531. Intermediate result 631 is delayed by register 533 to output signal 641 in S_{A2} on line A2 (523) at time t_2 603, after which signal I_1 615 is again assigned slot 623 in S_I as input for multiplier 223. At time t_3 604, the result of adder 235 is output from registers 535 to generate on line B1 (525) signal 651 in S_{B1}, which is similarly delayed by register 537 to output on line B2 (527) signal 661 in S_{B2} at time t_4 605, after which signal I_1 615 is again assigned slot 625 in S_I as input for multiplier 225. At time t_5 606, signal 671 of S_O on line O (509 in **Figure 5**), the result from adder 237 for input signal I_1 615, is ready for output by de-multiplexer 513. Similarly, signal 672 is the result of input signal I_2 617, computed from the input assigned to slots 622, 624 and 626. Further, the intermediate results from previous inputs are pipelined in the system so that results based on the previous inputs are available in the intermediate next clock cycles (e.g., results based on samples 623, 625 and 627 of input I_1 is available at time slot 673).

[0080] Thus, **Figures 5** and **6** illustrate that, when a register in the single-channel design is replaced by a cascade of registers, the intermediate result stored in the register of the single-channel design is pipelined in the cascade of registers in the multi-channel design so that the output the cascade of registers is synchronized with the time slot assignment for the signal processing in the multi-channel circuit. In one embodiment of the present invention, each of the channel-specific sequential elements (e.g., registers, flip-flops, memory elements) in the single-channel design is replaced with corresponding elements of N-times more capacity (e.g., a cascade of registers or flip-flops, dual-port RAM addressed according to the time slot

assignment, RAM-shift register combo, and others).

[0081] **Figure 7** shows an example of a multi-channel filter automatically generated from the single-channel filter of **Figure 2** according to one embodiment of the present invention. To automatically convert single-channel design 200 of **Figure 2** into time multiplexed N-channel design 700 of **Figure 7**, register 231 of **Figure 2** is replaced with a cascade of N-shift registers 711 – 719; and, register 233 of **Figure 2** is replaced with a cascade of N-shift registers 721 – 729. Modulo-N counter 705 is used to generate a signal for controlling the time slot assignment for input signals. When the output of modulo-N counter 705 on line 707 is i (i=0,1, ..., N-1), multiplexer 701 selects signal I_{i+1} as the input signal to line 709. Similarly, the output signal from adder 237 in **Figure 7** is decoded by de-multiplexer 703 to generate output signals for corresponding channels according to the output of modulo-N counter 705.

[0082] **Figure 8** shows an example of an output decoder and latch circuit for de-multiplexing outputs from a multi-channel filter according to one embodiment of the present invention. Testers 811, 813, ..., 819 control latches 801, 803, ..., 809 according to the state of selection signal 841 (e.g., from modulo-N counter 705 of **Figure 7**). The signal on line 843 (e.g., the output of adder 237 of **Figure 7**) is latched on one of output lines 831, 833, ..., 839, when the state of the selection signal matches the corresponding one of the constants (821 – 829) for the testers (811 – 819).

[0083] **Figure 9** shows another example of a multi-channel filter automatically generated from the single-channel filter of **Figure 2** according to an alternative

embodiment of the present invention. In **Figure 9**, circuit 200 is accessed in a round-robin fashion to process each of the input signals. Modulo-3 counter 907 allows input signals for each of the channels to be completely processed to generate an output signal before the channel is used for the processing of the signals of the next channel. Modulo-N counter 905 selects the signals of the channel to be processed by circuit 200. Although the method of **Figure 9** makes fewer modifications to the single-channel circuit in generating the multi-channel design, the circuit of **Figure 9** has a smaller throughput than the circuit of **Figure 7**. The signal pipelining in the single-channel design is not fully utilized in **Figure 9** because of the round-robin scheme. Depending on the structure of the input and output signals, the latency for the processing of signals for each channel may be minimized when a multi-channel design of **Figure 9** is used. However, it is understood that, in general, the input signals for different channels do not arrive in a round-robin fashion; instead, the input signals for different channels arrive at the same time. To retiming the input signals so that the input signals arrive in a round-robin fashion, a buffer can be used between the input signal and multiplexer 901. Such a buffer samples input signals at the same time, but delays the input signals from different channels for different amount of time so that the sampled signals arrived at multiplexer 901 in a round-robin fashion.

[0084] It is noticed that the time-multiplexed multi-channel of **Figure 7** can be used directly to replace the conglomerate of single-channel systems (e.g., in **Figure 4**) when the frequency of the clock signal C_N (741) is N times the frequency of the clock signal of the single-channel systems (e.g., clock signal C_S 431 of **Figure 4**).

When a design of **Figure 9** is used, an additional circuitry (e.g., a block of RAM with addressed according to the timing of the input signals and the time slot assignment for processing) can be used to retime the inputs.

[0085] In one embodiment of the present invention, each of the channel-specific elements (e.g., registers, constants, ROM or RAM) of the single-channel design is replaced with corresponding elements to pipeline the processing for multi-channel inputs. Although each of the registers (or flip-flops) can be considered a channel-specific register, which is replaced with a cascade of shifting registers, pipeline registers can be identified as non-channel-specific registers. When the pipeline registers are not replaced with cascade of shifting registers, the timing within the time-multiplexed shared channel can still be synchronized with respect to the input of the channels.

[0086] A set of pipeline registers is a feed-forward cutset of the system graph. As a feed-forward cutset of the system, the removal of the set of pipeline registers partitions the system into two disconnected subsystems, with a unidirectional flow of data between the two subsystems. However, when there are multiple inputs and/or multiple outputs in the system, the cutset that partitions the inputs into different subsystems or the outputs into different subsystems does not qualify as a set of pipeline registers, since synchronization can be distorted if such a cutset is not pipelined in generating the multi-channel design.

[0087] Feed-forward cutsets can be automatically identified using various algorithms known in the art. For example, Eran Halperin and Uri Zwick described methods in "Combinatorial approximation algorithms for the maximum directed cut

problem", Proceedings of 12th Symposium on Discrete Algorithms, pp. 1-7, 2001, which can be used to determine the feed-forward cutsets. It is noticed that feed-forward cutsets are often referred to as "directed cuts" (or "dicuts") in graph theory literature. The methods of Eran Halperin and Uri Zwick for finding "maximum directed cuts" can be used to find the feed-forward cutsets containing maximum numbers of registers.

[0088] **Figure 10** shows an example of a single-channel three-tap Finite Impulse Response (FIR) filter with pipeline registers from which a multi-channel filter can be automatically generated according to one embodiment of the present invention. The design of **Figure 10** includes pipeline registers 1001 – 1005. Feed-forward cutset 1011 partitions the system of **Figure 10** into two subsystems with a unidirectional flow of data between the two subsystems. Thus, registers 1001 – 1005 are identified as non-channel-specific; and thus, the channel-specific elements in the design of **Figure 10** are registers 231 and 233. Channel specific registers 231 and 233 in **Figure 10** are replaced with cascades of registers to generate time-multiplexed resource shared design of **Figure 11**.

[0089] **Figure 11** shows an example of a multi-channel filter automatically generated from the single-channel filter of **Figure 10** according to one embodiment of the present invention. In **Figure 11**, pipeline registers 1011 remain unchanged. To synchronize the operation of the de-multiplexer 703 with the state of Modulo-N counter 705, register 1201 is inserted to delay the output of counter 705 for one cycle to offset the effect of the delay caused by cutset 1011. In general, when M sets of pipeline registers are identified as non-channel specific elements, the output of the

Modulo-N counter is delayed mod(M, N) cycles to synchronized the operations of multiplexer and de-multiplexer (e.g., 701 and 703 in **Figure 11)**. It is also notice that registers 231, 1003 and 1005 in **Figure 10** can be identified as a set of pipeline registers, in which case registers 1001 and 233 will be replaced with cascades of shifting registers. Alternatively, if none of the registers in **Figure 10** is identified as pipeline registers, all registers in **Figure 10** are replaced with cascades of registers, in which case the resulting design will have N sets of pipeline registers and no register is necessary to delay the output of modulo-N counter 705, since mod(N,N)=0.

[0090] A channel-specific register of the single-channel system can be replaced with a cascade of N-shift registers in generating the multi-channel system. In some FPGA architectures (such as Xilinx Virtex), shift registers are natural primitives. This enables very efficient memory usage when the target architecture is one of these FPGA architectures. It is understand that each of the channel-specific registers can be also be replaced by other memory structures (e.g., an N-item dual-port RAM, or a RAM-shift register combo, or others) that can pipeline the intermediate results according to the state of the module-N counter.

[0091] **Figure 12** shows another example of a single-channel circuit with pipeline registers from which a multi-channel filter can be automatically generated according to one embodiment of the present invention. The design of **Figure 12** contains feed-forward cutsets 1211 and 1213, including registers 1201, 1203, 231 and 1205, 233. Thus, all registers in **Figure 12** can be identified as non-channel specific; and, no register is replaced with a cascade of registers in generating the

design of **Figure 13**.

[0092] **Figure 13** shows an example of a multi-channel filter automatically generated from the single-channel design of **Figure 12** according to one embodiment of the present invention. Since there are two sets of pipeline registers between multiplexer 1301 and de-multiplexer 1303, two cycles of delay can be used to synchronize the operations of the multi-channel filter. The design of **Figure 13** rotates the constants of de-multiplexer 1303 for two shifts to compensate the delay caused by the two sets of pipeline registers. Thus, when modulo-N counter outputs i ($i=0,1, \ldots, N-1$), multiplexer 1301 selects signal I_{i+1} as the input while de-multiplexer 1303 outputs for $O_{mod(i-1,N)}$.

[0093] From the above examples, it will be apparent to one skilled in the art that different methods (e.g., inserting delay elements, shifting constants for the de-multiplexer, generating different selection signals, or combination of these) can be used to compensate the delays caused by the sets of pipeline registers that remain unchanged in the time-shared design. Further, in general, a Finite State Machine (FSM) (e.g., a modulo-N counter) can be used to control the time multiplexing of the input signals, as well as the resource sharing in the design.

[0094] Although the above examples are illustrated using a single-channel design with a single input and a single output, from this description, it will be apparent to one skilled in the art that methods of various embodiments of the present invention can also be applied to a single-channel design with multiple inputs and multiple outputs. Further, an M-channel design ($M > 1$) can be treated as a single-channel design to automatically generate an N×M-channel design with resource

sharing.

[0095] In one embodiment of the present invention, the single-channel design is optimized before and/or after the automatic transformation in generating the resource shared design for multiple channels using conventional methods (e.g., folding transformation, and others).

[0096] **Figure 14** shows a flow chart of a method to generate a multi-channel circuit from a single-channel circuit according to one embodiment of the present invention. After a single-channel design is received in operation 1401, operation 1403 automatically transforms the single-channel design to generate a time multiplexed multi-channel design. Since the transformation is based on a single-channel design, which has fewer logical elements than a conglomerate of single-channel systems, computational intensive operations of extracting high level parallelism are avoided.

[0097] **Figure 15** shows a detailed flow chart of a method to generate a multi-channel circuit from a single-channel circuit according to one embodiment of the present invention. Operation 1501 receives a design of a single-channel circuit. Operation 1503 generates an N-state finite-state-machine (FSM) (e.g., a Modulo-N counter) to time multiplex access to the logic elements of the single-channel circuit. Operation 1505 generates a multiplexing circuit to multiplex, according to the state of the FSM, N-channel inputs as the input to the single-channel circuit. Operation 1507 replaces each channel-specific element (e.g., RAM, ROM, constants, registers, flip-flops) of the single-channel circuit with corresponding elements that are accessed for multiple channels according to the state of the FSM. Operation 1509

generates a de-multiplexing circuit to de-multiplex, according to the state of the FSM, from the corresponding output of the single-channel circuit into N-channel outputs. After the above transformation, a multi-channel design is generated from the single-channel design.

[0098] **Figure 16** shows an example method to generate a multi-channel circuit from a single-channel circuit according to one embodiment of the present invention. Operation 1601 receives a design of a single-channel circuit. Operation 1603 generates a modulo-N counter to control the logic elements of the single-channel to perform operations for the signal of channel i (i=0, 1, ..., N-1) when the value in the counter is i. Operation 1605 generates an N-item multiplexer to receive the input for channel i as the input for the single-channel circuit when the value in the counter is i. Operation 1607 optionally identifies non-channel-specific sequential elements (e.g., flip-flops, registers, ROM, constant, RAM) in the single-channel circuit. Operation 1609 replaces each of the channel-specific registers (e.g., flip-flop) of the single-channel design with N sequential elements (e.g., a cascade of N-shift registers, an N-item dual-port RAM, or RAM-shift register combo). Operation 1611 replaces each channel-specific memory (e.g., a RAM or ROM) of size M by a new memory of size $N \times M$ where memory item $N \times j + i$ of the new memory is addressed for channel i when item j of the memory is addressed in the single-channel circuit. Operation 1613 replaces each channel-specific constant with an N-item ROM memory where memory item i is addressed for the constant of channel i. Operation 1615 generates an N-item de-multiplexer to generate output for each channel from the corresponding output of the single-channel circuit.

[0099] In one embodiment of the present invention, a retiming algorithm (e.g., cutset retiming and pipelining, retiming for clock period minimization, such as those described in "VLSI digital signal processing systems: design and implementation", by Keshab K. Parhi, Wiley-Interscience, 1999, pp. 91-118, or others known in the art) is further used to improve the clock rate of the circuit, using the registers introduced during the automatic generation of the multi-channel circuit.

[00100] During the process of a circuit design, a negative latency register, which has a negative delay, can be used in an intermediate stage (e.g., in peripheral retiming, or in architectural retiming). While no physical negative latency register exists, it is understood that a negative latency register indicates a timing constraint at an intermediate state of design. Negative latency registers can be used in the early stage of the design process; and the negative latency registers are typically eliminated during the process of circuit design to achieve a final solution.

[00101] In one embodiment of the present invention, a single-channel system is specified with one or more negative latency registers; and, the negative latency registers can be transformed in a similar fashion as the regular registers.

[00102] **Figure 17** shows an example of a single-channel system. For the purpose of illustration, the latency of elements other than registers is ignored. Each register (e.g., 1731, 1733 and 1741) has a one-unit latency. Elements 1711, 1713 and 1715 are constants, which do not change in time. Thus, assuming the input on line 1701 is $I_1(t)$, the signal on line 1703 is $I_1(t) \times C_3 + I_1(t-2) \times C_2 + I_1(t-2) \times C_1$. In **Figure 17**, register 1731 stores the intermediate result from multiplier 1721 to generate $I_1(t-1) \times C_1$; and, register 1733 stores the intermediate result from

multiplier 1723 to generate $I_1(t-1) \times C_1$. Register 1741 stores the intermediate result from adder 1751 to generate $I_1(t-2) \times C_2 + I_1(t-2) \times C_1$.

[00103] A pair of positive and negative latency registers can be inserted into path between multiplier 1725 and adder 1753. After the insertion, the circuit in **Figure 17** is transformed to that in **Figure 18**. In **Figure 18**, register 1743 is a negative latency register; and, register 1735 is a regular register (positive latency register). It is seen that in **Figure 18** registers 1731, 1733 and 1735 is a set of pipeline register. Note that the removal of the set of pipeline register (1731, 1733 and 1735) changes the latency of the signal-channel circuit (which may be acceptable or desirable in some instances); alternatively, a register (not shown in **Figure 19**) can be inserted before point 1701 in **Figure 19** to have a single-channel circuit that has the same latency as that of the circuit in **Figure 18**. Thus, the single-channel circuit in **Figure 18** can be specified as that in **Figure 19**. In **Figure 19**, assuming the input on line 1701 is $I_1(t)$, the output on line 1703 is $I_1(t+1) \times C_3 + I_1(t-1) \times C_2 + I_1(t-1) \times C_1$. Thus, apart from a one-unit timing shift, the circuit in **Figure 19** performs essentially the function as the circuit in **Figure 17** (or **Figure 18**). When the circuit in **Figure 19** is specified as an input, a multi-channel circuit can be automatically generated. Registers 1741 and 1743 are not pipeline registers; and, they can be replace with multiple cascaded registers. **Figure 20** shows an example of a two-channel circuit automatically generated from the input of **Figure 19**. Negative latency register 1743 in **Figure 19** is replaced with cascaded negative latency registers 1771 and 1773; and, register 1741 in **Figure 19** is replaced with cascaded registers 1761 and 1763. Multiplexer 1705 is added to feed the input signals into the shared channel one at a

time; and, demultiplexer 1707 is added to restore the output from the shared channel one at a time. After the generation of the multi-channel circuit, a retiming algorithm (e.g., those described in “VLSI digital signal processing systems: design and implementation”, by Keshab K. Parhi, Wiley-Interscience, 1999, pp. 91-118, or others known in the art) can be used to optimize the system. For example, pipeline register sets can be inserted into eliminate the negative latency registers. For example, two pipeline register sets can be inserted after multipliers 1721, 1723 and 1725 to generate the circuit in **Figure 21**. Note that, as described previously, the insertion or deletion of pipeline register sets can change the timing for the generation of output on line 1703 in general; and thus, a proper number of registers (or delay elements) can be used on line 1709 to adjust the timing of the control signal for the demultiplexer 1707 (as illustrated in **Figure 11**), the correspondence between the control signal and the output line can be adjusted (as illustrated in **Figure 13**). It is seen that the circuit in **Figure 21** can be generated directly from the circuit of **Figure 17**, according to embodiments of the present invention. Note that the two pipeline register sets can also be inserted after multipliers 1725 and adder 1751 in **Figure 20** to eliminate negative latency registers.

[00104] Further, in one embodiment of the present invention, one or more pairs of positive and negative latency registers are introduced into the single-channel system in the process of identifying pipeline registers. For example, after the single-channel circuit of **Figure 17** is received for the generation of multi-channel circuit, a positive and negative latency register pair (e.g., 1735 and 1743 in **Figure 18**) is inserted for identifying pipeline registers. As illustrated in **Figure 18**, once regular

register 1735 and negative latency register 1743 are inserted, pipeline register sets (1731, 1733 and 1735) can be identified. Thus, only registers 1741 and 1743 of **Figure 18** are replaced with corresponding cascaded registers. Similarly, retiming algorithms can be used to further optimize the circuit of the automatic generation of the multi-channel circuit.

[00105] **Figures 22 – 25** illustrate another example of generating a multi-channel circuit from a single-channel circuit using negative latency registers according to one embodiment of the present invention. The single-channel filter of **Figure 22** has pipelined adder 1847 (A_2) and pipelined multipliers 1841-1843 (M_1 -M_3). A pipelined adder has an embedded register. For example, pipelined adder 1847 has register 1865, which is a physical part of the pipelined adder and cannot be moved out. Similarly, a pipelined multiplier also has an embedded register (e.g., pipelined multipliers 1841, 1842 and 1843 has registers 1831, 1833 and 1835 respectively).

[00106] Consider that a pipelined adder is to be used to implement adder 1851. Without using a negative latency register, a set of regular registers may be added (e.g., on both the input lines for adder 1853) to provide adder 1851 a register. Such an approach can lead to the increase of the latency of the single channel system. Alternatively, **Figure 23** shows an example in which a pair of positive and negative latency registers (1861 and 1863) are added between adders 1851 and 1853. Thus, adder 1851 and register 1861 can be implemented as a pipelined adder 1845 (A_1). Note that negative latency register cannot be eliminated in the single-channel system even with retiming.

[00107] A multi-channel circuit can be automatically generated according to

embodiments of the present invention. For example, **Figure 23** shows the time-shared portion of a multi-channel circuit, generated according to one embodiment of the present invention for a two-channel circuit. Note that the input multiplexing portion and the output de-multiplexing portion of the two-channel circuit are not shown in **Figure 24**. Registers 1831, 1833 and 1835 are channel specific in **Figure 23**; and, registers 1832, 1834 and 1836 are inserted in **Figure 24** for the multi-channel circuit. Similarly, registers 1861 and 1863 in **Figure 23** are replaced with cascaded register sets (1861,1862 and 1863 and 1864) in **Figure 24**.

[00108] In the two-channel circuit generated according to one embodiment of the present invention, extra registers are generated for each of the pipelined operators. And, a retiming operation can be used to remove the negative registers. For example, registers 1832 and 1834 in **Figure 24** can be moved to the path between adders 1851 and 1853 in a retiming operation so that there are sufficient regular registers on the path between adders 1851 and 1853 to cancel out the negative registers (1863 and 1864). Thus, an example resulting two-channel circuit is shown in **Figure 25**. In **Figure 25**, multipliers 2821, 1823 and 1825 and adders 1851 and 1853 can be implemented as corresponding pipelined operators (pipelined multipliers 1841, 1842 and 1843 and pipelined adders 1845 and 1847). However, all negative latency registers are eliminated after retiming.

[00109] Pipelined adders and pipelined multipliers are illustrated in the above example. However, from this description, it will be appreciated that other pipelined logic elements or pipelined IP blocks can also be treated in a similar fashion.

[00110] In one embodiment of the present invention, no pipeline register sets are

identified; and, all registers are considered as channel-specific. For example, an automatic method to transform a single-channel system into an N-channel system includes the following operations.

[00111] 1. Generate a modulo-N counter as an N-state finite-state-machine (FSM). The state variable of the FSM is denoted as cnt. The m'th channel ($0 \leq m \leq N-1$) of the system is effectively activated when the FSM is in the m'th state (e.g., when cnt = m).

[00112] 2. Replace each register by a cascade of N-shift registers (or alternatively, if N is large, each register can be replaced with an N-item dual-port RAM or a RAM-shift register combo).

[00113] 3. Replace each RAM memory of size M by a RAM memory of size N×M. The address generation circuitry is modified or added to address the memory items such that memory item A of the single-channel design is replaced by an item at address N×A+cnt. It is understood that other address transformation scheme can also be used to logically pipeline memory items so that the output from the new memory is synchronized with the state of the FSM. The transformed address is typically a function of the original address, the state of the FSM, and the active channel number.

[00114] 4. When there is a channel-specific ROM (e.g., ROM contents vary from channel to channel), apply a transform similar to that for the RAM so that ROM item N×A+m holds the contents of the ROM item A for m'th channel. If the ROM is not channel specific, no transformation is necessary.

[00115] 5. If there is a channel-specific constant (e.g., constant value varies from

channel to channel), replace it with an N-item ROM. The address line of the ROM is driven by cnt (or cnt – 1 if there is one clock latency in ROM access). If the constant is not channel specific, no transformation is necessary.

[00116] The inputs of the different channels may be multiplexed over a common port. If multiple channels have their distinct input ports, multiplex these inputs by an N-item multiplexer with a selection line driven by cnt, which is N-times faster than the clock driving the inputs. Similarly, the outputs of the different channels may be multiplexed over a common port. If multiple channels have their distinct output ports, de-multiplex the outputs by an N-item decoder with a selection line driven by cnt. The outputs can be latched with an N-times slower clock. The other components of the single-channel design are kept as is as the shared resources through time multiplexing.

[00117] In another embodiment of the present invention, pipeline register sets are identified to avoid the generation of registers. For example, an automatic method to transform a single-channel system into an N-channel system includes the following operations.

[00118] 1. Identify the pipeline registers in the system to cover as many and as wide registers as possible. There can be more than one set of pipeline registers, but the sets must be mutually exclusive. Any register not classified into any of the pipeline register sets is a state register. Pipeline registers are non-channel-specific; and, state registers are channel-specific. If there are P distinct pipeline register sets, the system is partitioned into P+1 disjoint subsystems, forming a singly linked list of subsystems. In this list, each feed-forward cutset defines a link, with the direction of

the link indicating the direction of data flow. Denote S_i the subsystem whose order in the linked list is i, where $0 \leq i \leq P$.

[00119] 2. Generate P+1 N-state finite-state-machines (FSMs) from at least one modulo-N counter. Denote the state variable of i'th FSM ($0 \leq i \leq P$) as cnt_i. The hardware in subsystem S_i processes the data of the m'th channel ($0 \leq m \leq N-1$) when the i'th FSM is in the m'th state (e.g., when $cnt_i = m$). When $mod((cnt_i - cnt_j),N) = mod((j-i), N)$ for $\forall$ i , j in [0,P], correct synchronization is maintained among the subsystems. This relation is satisfied if cnt_i is a one-clock delayed version of cnt_{i-1}. (i=1,2, …, N-1). Thus, cnt_i can be derived from cnt_{i-1} using a register; and, the FSMs can be implemented using one modulo-N counter and a cascade of registers.

[00120] 3. Replace each state register with a cascade of N-shift registers (or alternatively, if N is large, each state register can be replaced with an N-item dual-port RAM or a RAM-shift register combo).

[00121] 4. Replace each RAM memory of size M by a RAM memory of size N×M. For a RAM within the subsystem S_i, the address generation circuitry is modified or added to address the memory items such that memory item A of the single-channel design is replaced by an item at address $N \times A + cnt_i$. It is understood that other address transformation scheme can also be used to logically pipeline memory items so that the output from the new memory is synchronized with the state of the i'th FSM. The transformed address is typically a function of the original address, the state of the i'th FSM, and the active channel number.

[00122] 5. When there is a channel-specific ROM (e.g., ROM contents vary from channel to channel), apply a transform similar to that for the RAM so that ROM

item N×A+m holds the contents of the ROM item A for m'th channel. If the ROM is not channel specific, no transformation is necessary.

[00123] 6. If there is a channel-specific constant (e.g., constant value varies from channel to channel) within the subsystem S_i, replace it with an N-item ROM. The address line of the ROM is driven by cnt_i (or $cnt_i - 1$ if there is one clock latency in ROM access). If the constant is not channel specific, no transformation is necessary.

[00124] The inputs of the different channels may be multiplexed over a common port. If multiple channels have their distinct input ports, multiplex these inputs by an N-item multiplexer with a selection line driven by cnt_0, which is N-times faster than the clock driving the inputs. Similarly, the outputs of the different channels may be multiplexed over a common port. If multiple channels have their distinct output ports, de-multiplex the outputs by an N-item decoder with a selection line driven by cnt_P. The outputs can be latched with an N-times slower clock. The other components of the single-channel design are kept as is as the shared resources through time multiplexing.

[00125] Although P+1 N-state finite-state-machines can be used to control the timing of the P+1 subsystems individually, alternatively, one single N-state FSM can be used to control the operation of all subsystems, where different subsystems process for a given channel when the state of the FSM reaches different values. For example, the hardware in subsystem S_i processes the data of the m'th channel ($0 \leq m \leq N-1$) when the FSM is in the state cnt = mod (m-i, N).

[00126] Pipeline register sets can be identified and removed, when timing constraints permit. For example, the set of pipeline registers 1011 in **Figure 10** can

be removed to generate the design of **Figure** 2; or, the pipeline register set 1011 of **Figure 11** can be removed automatically to generate the design of **Figure 7**. Similarly, pipeline registers can also be added to a design automatically. Note that the addition or deletion of pipeline registers in general changes input to output latency of the circuit; however, such changes are acceptable for most cases.

[00127] In one embodiment of the present invention, employing shift registers with run-time-configurable depth and using modulo-N counters with run-time-configurable modulus, the folding rate N can be changed in run-time, without interrupting the operation. For the applications in which the number of active channels is a dynamic parameter, the folding rate can be adapted to the number of active channels, allowing the clock rate to be reduced, when possible, to reduce the power consumption.

[00128] Thus, at least one embodiment of the present invention automatically generates a design for a multi-channel system from the input of the design of single-channel system. The single-channel system design is automatically transformed into an N-channel system with time-multiplexed resource sharing of logical (computational) units. The transform is simple and very fast; and, the resulting design of the hardware is very efficient.

[00129] Since methods of various embodiment of the present invention generate a time-multiplexed multi-channel design from a single-channel design, fast optimization and synthesis operations can be performed on the reduced number of logic elements. The usage of control logic can be minimized. For some FPGA architectures, the memory usage of the synthesized system can be made very

efficient. Further, various methods of the present invention allow for high pipeline orders and significant speed-ups.

[00130] Many of the methods of the present invention may be performed with a digital processing system, such as a conventional, general-purpose computer system. Special purpose computers, which are designed or programmed to perform only one function, may also be used.

[00131] **Figure 1** shows one example of a typical computer system which may be used with the present invention. Note that while **Figure 1** illustrates various components of a computer system, it is not intended to represent any particular architecture or manner of interconnecting the components as such details are not germane to the present invention. It will also be appreciated that network computers and other data processing systems which have fewer components or perhaps more components may also be used with the present invention. The computer system of **Figure 1** may, for example, be a Sun workstation, or a personal computer (PC) running a Windows operating system, or an Apple Macintosh computer.

[00132] As shown in **Figure 1**, the computer system 101, which is a form of a data processing system, includes a bus 102 which is coupled to a microprocessor 103 and a ROM 107 and volatile RAM 105 and a non-volatile memory 106. The microprocessor 103, which may be a G3 or G4 microprocessor from Motorola, Inc. or IBM is coupled to cache memory 104 as shown in the example of **Figure 1**. The bus 102 interconnects these various components together and also interconnects these components 103, 107, 105, and 106 to a display controller and display device 108 and to peripheral devices such as input/output (I/O) devices which may be mice,

keyboards, modems, network interfaces, printers, scanners, video cameras and other devices which are well known in the art. Typically, the input/output devices 110 are coupled to the system through input/output controllers 109. The volatile RAM 105 is typically implemented as dynamic RAM (DRAM) which requires power continually in order to refresh or maintain the data in the memory. The non-volatile memory 106 is typically a magnetic hard drive or a magnetic optical drive or an optical drive or a DVD RAM or other type of memory systems which maintain data even after power is removed from the system. Typically, the non-volatile memory will also be a random access memory although this is not required. While **Figure 1** shows that the non-volatile memory is a local device coupled directly to the rest of the components in the data processing system, it will be appreciated that the present invention may utilize a non-volatile memory which is remote from the system, such as a network storage device which is coupled to the data processing system through a network interface such as a modem or Ethernet interface. The bus 102 may include one or more buses connected to each other through various bridges, controllers and/or adapters as is well known in the art. In one embodiment the I/O controller 109 includes a USB (Universal Serial Bus) adapter for controlling USB peripherals, and/or an IEEE-1394 bus adapter for controlling IEEE-1394 peripherals.

[00133] It will be apparent from this description that aspects of the present invention may be embodied, at least in part, in software. That is, the techniques may be carried out in a computer system or other data processing system in response to its processor, such as a microprocessor, executing sequences of instructions contained in a memory, such as ROM 107, volatile RAM 105, non-volatile memory

106, cache 104 or a remote storage device. In various embodiments, hardwired circuitry may be used in combination with software instructions to implement the present invention. Thus, the techniques are not limited to any specific combination of hardware circuitry and software nor to any particular source for the instructions executed by the data processing system. In addition, throughout this description, various functions and operations are described as being performed by or caused by software code to simplify description. However, those skilled in the art will recognize what is meant by such expressions is that the functions result from execution of the code by a processor, such as the microprocessor 103.

[00134] A machine readable medium can be used to store software and data which when executed by a data processing system causes the system to perform various methods of the present invention. This executable software and data may be stored in various places including for example ROM 107, volatile RAM 105, non-volatile memory 106 and/or cache 104 as shown in **Figure 1**. Portions of this software and/or data may be stored in any one of these storage devices.

[00135] Thus, a machine readable medium includes any mechanism that provides (i.e., stores and/or transmits) information in a form accessible by a machine (e.g., a computer, network device, personal digital assistant, manufacturing tool, any device with a set of one or more processors, etc.). For example, a machine readable medium includes recordable/non-recordable media (e.g., read only memory (ROM); random access memory (RAM); magnetic disk storage media; optical storage media; flash memory devices; etc.), as well as electrical, optical, acoustical or other forms of propagated signals (e.g., carrier waves, infrared signals, digital signals, etc.); etc.

[00136] In the foregoing specification, the invention has been described with reference to specific exemplary embodiments thereof. It will be evident that various modifications may be made thereto without departing from the broader spirit and scope of the invention as set forth in the following claims. The specification and drawings are, accordingly, to be regarded in an illustrative sense rather than a restrictive sense.